BAB 2
LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dalam kajian pustaka ini menjelaskan tentang dasar atau acuan berupa teori-teori atau temuan-temuan melalui hasil dari berbagai penelitian terdahulu yang relevan dengan permasalahan yang sedang dibahas dan dijadikan oleh penulis sebagai referensi dan data pendukung dalam penelitian ini, sekaligus menjadi bukti bahwa penelitian ini layak untuk dilakukan. Fokus penelitian terdahulu yang dijadikan acuan dalam hal ini adalah terkait dengan masalah teknologi informasi. Oleh sebab itu, peneliti melakukan langkah kajian terhadap beberapa hasil penelitian berupa tesis atau jurnal-jurnal yang relevan dengan penelitian ini yang dirangkum dalam Tabel 2.1

Tabel 2.1 Kajian Hasil Penelitian Sebelumnya

No	Tahun	Peneliti	Judul	Hasil/Temuan	Variabel Terkait
1	2014	Refli Agustiawan	*"WEB-GIS Penentuan Potensi Usaha Menggunakan Metode Analytical Hierarchy Process (AHP)"*	WEB-GIS menjadi sebuah solusi untuk memetakan, menganalisis dan menentukan wilayah yang berpotensi usaha pada sektor pertanian, kesehatan, energi dan pertambangan secara matematis di Kabupaten Bolaang Mongondow	Memaparkan data spasial untuk lokasi penentuan potensi usaha
2	2011	Hanafi	*"SIG dan AHP Untuk Sistem Pendukung Keputusan Perencanaan Wilayah*	Sistem Pendukung Keputusan Berbasis SIG yang dapat	Memaparkan penentuan lokasi menggunakan metode *AHP* pada sistem

			Industri dan Pemukiman Kota Medan"	membantu perencana dalam menentukan lokasi untuk wilayah industri dan pemukiman di Kota Medan. Namun penelitian ini tidak menganalisis lokasi-lokasi yang berpotensi usaha	pemetaan berbasis WEB-*GIS*
3	2015	Alfian Pratama Putra	*"Rancang Bangun Sistem Informasi Geografis (GIS) Potensi Hasil Perikanan Di Kota Pasuruan (Studi Kasus Dinas Perikanan Kota Pasuruan)"*	Memberikan infromasi tentang wilayah pesisir pantai yang memiliki potensi hasil perikanan di Kota Pasuruan yang dikemas dalam peta berbasis WEB-*GIS*	Memaparkan rekomendasi potensi hasil perikanan unggulan dalam bentuk data spasial berbasis WEB-*GIS*

2.1.1 Penelitian Terkait

Terdapat beberapa penelitian sebelumnya terkait dengan Analisis Penentuan Potensi Usaha menggunakan Metode Analytical Hierarchy Process (AHP) Berbasis WebGis, namun yang lebih mendekati dari penelitian ini adalah penelitian yang dilakukan oleh Hanafi (2011), yang menghasilkan Sistem Pendukung Keputusan Berbasis SIG yang dapat membantu perencana dalam menentukan lokasi untuk wilayah industri dan pemukiman di Kota Medan. Namun penelitian ini tidak menganalisis lokasi-lokasi yang berpotensi usaha.

Perbedaan penelitian di atas dengan penelitian yang penulis lakukan yaitu penulis menganalisis potensi lahan kemudian memetakannya dengan SIG untuk mengalokasikan wilayah yang berpotensi di bukanya usaha baik di bidang

Pertanian, Perikanan, Perkebunan, Kerajinan Umum dan Pariwisata berdasarkan dari kriteria masing-masing sesuai pada bidangnya.

2.2 Dasar Teori

Dasar teori yang mendukung pada penelitian pembangunan sistem pemetaan berbasis web-gis untuk Analisis potensi usaha di Kabupaten Malang menggunakan metode *Analytical Hierarchy Process* (AHP) adalah tentang Potensi Usaha, Rekayasa Perangkat Lunak, UML (*Unified Modeling Language)*, Pengujian Perangkat Lunak, Metode Pengembangan Perangkat Lunak, Metode *Analytical Hierarchy Process* (AHP), Peta dan Pemetaan, *Geographics Information Sistem (GIS)* dan *Google Maps API Library.*

2.2.1 Potensi Usaha

Definisi potensi adalah sesuatu hal yang dapat disajikan sebagai bahan atau sumber yang akan dikelola baik melalui usaha yang dilakukan manusia maupun yang dilakukan melalui tenaga mesin dimana dalam pengerjaannya potensi dapat juga diartikan sebagai sumber daya yang ada disekitar kita (Kartasapoetra, 2009). Menurut KBBI (Kamus Besar Bahasa Indonesia) potensi adalah kemampuan yang mempunyai kemungkinan untuk dikembangkan.

Potensi yang dibahas dalam penelitian ini yaitu sumber daya alam (SDA) yang dikelola dengan cermat oleh sumber daya manusia (SDM) dimana potensi SDA tersebut dapat menjadi satu kesatuan yang saling terintegrasi dalam pelaksanaan pembangunan suatu usaha yang ada di tingkat Kecamatan maupun Kabupaten. Khususnya Kabupaten Malang yang dikenal kawasan yang sangat memiliki banyak potensi tentunya potensi yang ada tersebut dapat dijadikan sebagai modal dalam pembangunan suatu usaha supaya masyarakatnya dapat memberdayakan potensi yang ada dengan sebaik mungkin demi meningkatkan kesejahteraan hidup mereka. Potensi sumber daya alam di Kabupaten Malang meliputi berbagai bidang diantaranya bidang Pertanian, Perikanan, Perkebunan, Kerajinan Umum dan Pariwisata.

2.2.2 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah suatu disiplin ilmu yang membahas semua aspek rekayasa perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan(Somad, A. 2011).

Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi criteria berikut(Somad, A. 2011):

- Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berkembangnya teknologi dan lingkungannya *(Maintainability)*
- Dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi *(Dependability dan Robust)*
- Efisien dari segi sumber daya dan pengunaan

- Kemampuan untuk dipakai sesuai dengan kebutuhan *(Usability)*

Dari kriteria di atas dapat disimpulkan bahwa perangkat lunak yang baik adalah perangkat lunak yang dapat memenuhi kebutuhan dari klien*(customer)* atau pegguna *(user)* atau berorientasi pada pengguna perangkat lunak, bukan berorientasi pada pembuat atau pengembang perangkat lunak itu sendiri.

Secara umum perekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun rekayasa ini sebenarnya mencakup masalah pemilihan metode yang paling sesuai untuk suatu keadaan, pendekatan yang paling kreatif, dan pengembangan yang mungkin efektif pada beberapa keadaan(Somad, A. 2011).

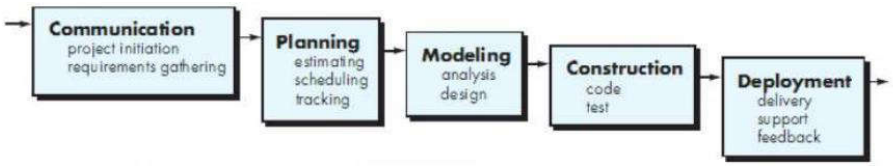
2.2.3 *Software Development Life Cycle (SDLC)*

Untuk mendapatkan *software* atau perangkat lunak yang berkualitas tinggi dan handal *(reliable)* dalam pengembangannya membutuhkan metode dan proses untuk mendapatkan hasil yang sesuai kebutuhan. Pengembangan perangkat lunak atau sistem itu harus terencana yang menggambarkan tahapan-tahapan yang sering disebut sebagai pendekatan *Sistem Development Life Cyle (SDLC)* (Tata, 2012).

Setiap *software engineer* harus bisa memahami dan mengikuti tahapan-tahapan yang telah ditetapkan di dalam *software engineer* [IEEE Standard 610.12]. Oleh karena itu, *software engineer* harus mampu memilih model proses perangkat lunak yang paling tepat sehingga mendapatkan kemudahan dalam mengelola dan mengendalikan proses pembangunan sistem, tahap demi tahap yang akan dilakukan.

2.2.3.1 Waterfall Model

Model waterfall adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Nama model ini sebenarnya adalah *"Linear Sequential Model".* Model ini sering disebut juga dengan *"classic life cycle"* atau metode waterfall. Model ini termasuk ke dalam model generic pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam Software Engineering (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman, 2015).



Gambar 2.1 *Waterfall Model*

Sumber: Pressman (2015)

Berikut ini merupakan penjelasan dari setiap tahapan yang terjadi dalam *Waterfall Model*:

a. *Communication (Project Initiation & Requirements Gathering)*

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan customer demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi software. Pengumpulan data-data tambahan bisa juga diambil dari beberapa sumber yang jelas dan dapat dipertanggungjawabkan.

b. Planning (Estimating, Scheduling, Tracking)

Berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang mungkin terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan tracking proses pengerjaan sistem.

c. Modeling (Analysis & Design)

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur software, tampilan interface, dan algoritme program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

d. Construction (Code & Test)

Tahapan Construction ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki

e. Deployment (Delivery, Support, Feedback)

Tahapan Deployment merupakan tahapan implementasi software ke customer, pemeliharaan software secara berkala, perbaikan software, evaluasi software, dan pengembangan software berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. (Pressman, 2015)

Dengan demikian, metode waterfall dianggap pendekatan yang lebih cocok digunakan untuk proyek pembuatan sistem baru dan juga pengembangan *software* dengan tingkat resiko yang kecil serta waktu pengembangan yang cukup lama. Tetapi salah satu kelemahan paling mendasar adalah menyamakan pengembangan hardware dan software dengan meniadakan perubahan saat

pengembangan. Padahal, *error* diketahui saat software dijalankan, dan perubahan-perubahan akan sering terjadi.

Jadi kelemahan menggunakan metode waterfall adalah bersifat kaku, sehingga sulit melakukan perubahan di tengah proses. Jika terdapat kekurangan proses/prosedur dari tahap sebelumnya, maka tahapan pengembangan harus dilakukan mulai dari awal lagi. Hal ini akan memakan waktu yang lebih lama. Karena jika proses sebelumnya belum selesai sampai akhir, maka proses selanjutnya juga tidak dapat berjalan. Oleh karena itu, jika terdapat kekurangan dalam permintaan user maka proses pengembangan harus dimulai kembali dari

Keuntungan menggunakan metode waterfall adalah prosesnya lebih terstruktur, hal ini membuat kualitas software baik dan tetap terjaga. Dari sisi user juga lebih menguntungkan, karena dapat merencanakan dan menyiapkan kebutuhan data dan proses yang diperlukan sejak awal. Penjadwalan juga menjadi lebih menentu, karena jadwal setiap proses dapat ditentukan secara pasti. Sehingga dapat dilihat jelas target penyelesaian pengembangan program. Dengan adanya urutan yang pasti, dapat dilihat pula perkembangan untuk setiap tahap secara pasti. Dari sisi lain, model ini merupakan jenis model yang bersifat dokumen lengkap sehingga proses pemeliharaan dapat dilakukan dengan mudah(Pressman, 2015).

2.2.4 Unified Modeling Language (UML)

UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML adalah notasi diagram yang menggunakan bahasa modeling berorientasi objek (Connolly dan Begg, 2010). Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML sintaks mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dai 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modelling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering) (Dharwiyanti. S, 2003)

Secara singkat dapat ditarik kesimpulan bahwa "*Unified Modelling Language (UML)"* adalah sebuah bahasa yang dikemas dalam bentuk grafik atau gambar untuk memvisualisasikan, menspesifikan, membangun dan mendokumentasikan dari sebuah sistem pengembangan perangkat lunak berbasis OO *(Object-Oriented)*".

Adapun *UML* sendiri dapat dideskripsikan dalam beberapa diagram, diantaranya: (Alfian, 2016)

1. Aktor

Tabel 2.2 Notasi Aktor

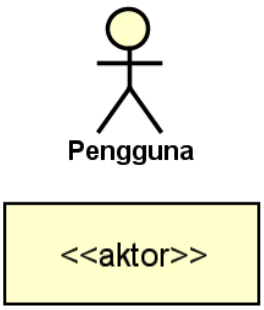
Nama	Notasi
Pengguna	
Administrator	

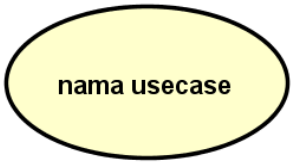
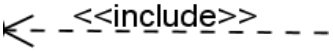
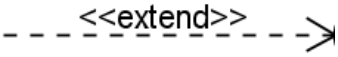
Pada dasarnya aktor bukanlah bagian dari *use case diagram,* namun untuk dapat terciptanya suatu *diagram use case* diperlukan sebuah aktor, dimana aktor tersebut merepresentasikan sesorang (manusia) atau sesuatu (seperti perangkat atau sistem lain) yang berinteraksi dengan sistem yang dibuat. Sebuah aktor mungkin hanya memberikan informasi masukan maupun menerima informasi dari sistem. (Alfian, 2016)

2. *Use Case Diagram*

Use Case Diagram menyajikan interaksi antara *use case* dan aktor, dimana aktor sendiri dapat berupa seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang dibangun. *Use Case* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari sudut pandang pengguna.

Tabel 2.3 Notasi *Use Case Diagram*

Nama	Deskripsi	Notasi
Aktor	- Seseorang atau sesuatu (sistem) yang memiliki peranan sebagai subjek - Diberi label sesuai dengan perannya - Digambarkan sebagai tongkat *(default),* jika aktor yang terlibat bukan seseorang maka digambarkan sebagai garis kotak persegi panjang dengan diapit tanda <<nama_aktor>> - Dapat diasosiasikan	

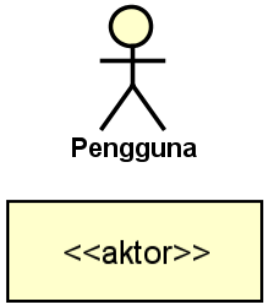
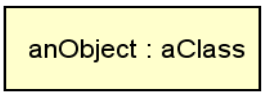
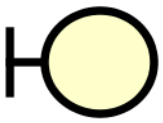
	dengan aktor lain menggunakan hubungan *specialization/superclass* - Diletakkan di luar boundary	
Use Case	- Merupakan bagian utama dari fungsi yang ada pada sistem - Diberi label nama usecase (kata kerja) - Dapat *extends* ke *use case* lainnya - Dapat *include* ke *use case* lainnya	
Assosiation Relationship	- Menghubungkan antara aktor dengan *use case*	
Include Relationship	- *Use Case* dapat melakukan *include* fungsionalitas *use case* lain sebagai dari proses dalam dirinya	
Extend Relationship	- *Use Case* dapat melakukan *extend use case* lain dengan *behaviour*-nya dirinya sendiri	
Generalization Relationship	- *Use Case* dapat melakukan *extend use case* lain dengan *behaviour*-nya dirinya sendiri	
Subject Boundary	- Merupakan ruang lingkup suatu sistem atau proses bisnis	

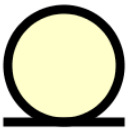
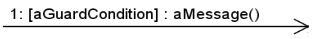
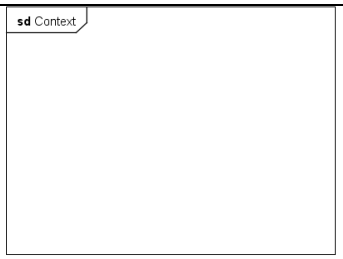
3. *Sequence Diagram*

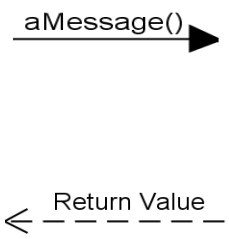
Sequence Diagram menggambarkan interaksi antar objek yang ada di dalam sistem maupun di sekitar sistem (termasuk pengguna,

display dan sebagainya) berupa *message* yang digambarkan berdasarkan waktu *(timeline).* Setiap objek yang terlibat dalam *use case diagram* digambarkan dengan garis putus-putus vertikal kemudian *message* yang dikirimkan oleh objek digambarkan dengan garis *horizontal* secara kronologis dari atas ke bawah. *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian untuk menghasilkan keluaran tertentu.

Tabel 2.4 Notasi *Sequence Diagram*

Nama	Deskripsi	Notasi
Aktor	- Digunakan sebagai aktor yang memulai alur peristiwa/kejadian - Ditempatkan di bagian atas diagram - Diberi label sesuai dengan perannya - Digambarkan sebagai tongkat *(default),* jika aktor yang terlibat bukan seseorang maka digambarkan sebagai garis kotak persegi panjang dengan diapit tanda <<nama_aktor>> - Berpartisipasi secara berurutan dengan mengirim dan menerima pesan	
General	- Merepresentasikan entitas tunggal dalam *sequence diagram,* digambarkan dengan garis kotak persegi panjang. - Entitas ini memiliki nama, *stereotype* atau berupa *instance [instance:Class]*	
Boundary	- Boundary biasanya berupa tepi dari sistem, seperti user interface atau suatu alat yang berinteraksi dengan	

	sistem lainnya	
Control	- Control element mengatur aliran dari informasi untuk sebuah skenario.	
Entity	- Entity biasanya elemen yang bertanggungjawab menyimpan data atau informasi. Ini dapat berupa *beans* atau *model object*	
Lifetime	- Menggambarkan hubungan suatu elemen yang berbeda - Berisi X pada titik dimana kelas tidak lagi berinteraksi	
Execution Occurrence	- Menunjukkan ketika suatu objek mengirim dan menerima *message* - Berbentuk kotak persegi panjang dan ditempatkan di atas lifetime	
Guard Condition	- Merupakan syarat yang dipenuhi untuk pesan yang dikirim	
Object Destruction	- Sebuah **X** ditempatkan pada akhir objek, untuk menunjukkan akan keluar eksistensi	**X**
Frame	- Menunjukkan context atau penamaan dari *sequence diagram*	
Message	- Menyampaikan informasi dari satu objek ke objek	

	lain - Panggilan operasi diberi label dengan pesan yang dikirim dengan arah panah - Return diberi label dengan nilai yang akan dikembalikan dan ditampilkan berupa arah panah dengan garis putus-putus	

4. *Class Diagram*

Class Diagram menunjukkan interaksi antar kelas yang ada dalam sistem. *Class Diagram* dibangun berdasarkan *use case diagram* dan *squence diagram* yang telah dibuat sebelumnya. *Class Diagram* terdiri dari tiga elemen utama didalamnya yaitu:

- Nama Kelas
- Atribut
- Operasi (*Method)*

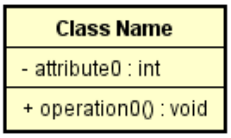
Atribut dan *method* sendiri dapat memiliki salah satu sifat berikut:

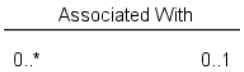
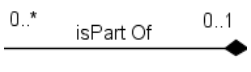
a. *Private,* tidak dapat dipanggil dari luar kelas yang bersangkutan.

b. *Protected,* hanya dapat dipanggil oleh kelas yang bersangkutan dan turunan kelas yang mewarisinya.

c. *Public,* dapat dipanggil oleh kelas apa saja.

Tabel 2.5 Notasi *Class Diagram*

Nama	Deskripsi	Notasi
Class	- Merepresentasikan sesorang tempat, atau hal yang berhubungan dengan sistem yang perlu di-*capture* dan menyimpan informasi - Nama kelas ditulis dengan huruf tebal dan berada di atas comparement - Terdiri dari list atribut dan operasi - Tidak secara eksplisit menunjukkan operasi yang	

	tersedia untuk semua kelas	
Attribute	- Menggambarkan keadaan atau objek - Bisa berasal dari atribut lainnya, yang ditunjukkan oleh garis miring sebelum nama atribut tersebut - Memiliki *attribute type* dan sifat sesuai dengan atribut yang diinginkan	
Operation	- Merupakan tindakan atau fungsi yang ada dalam kelas - Dapat diklasifikasikan sebagai konstruktor, query, atau operasi update - Tanda kurung "()" yang berisi parameter atau informasi yang diperlukan untuk melakukan operasi - Memiliki *operation type* dan sifat sesuai dengan operasi yang diinginkan	
Association	- Menggambarkan hubungan antara beberapa kelas atau kelas itu sendiri - Bisa di antara satu atau lebih kelas - Memilki label dengan frasa kata kerja atau nama peran - Berisi simbol yang merupakan batas minimum dan maksimum waktu kelas	
Aggregation	- Menggambarkan hubungan logis antar beberapa kelas atau kelas itu sendiri - Bentuk khusus dari hubungan asosiasi	
Composition	- Menggambarkan hubungan fisik antara beberapa kelas atau kelas itu sendiri	

	- Bentuk khusus dari hubungan asosiasi	
Generalization	- Menggambarkan hubungan turunan dari kelas	

2.2.5 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau biasanya dalam bahasa inggris disebut *Object-Oriented Programming (OOP)* merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Dibandingkan dengan logika pemrogrman terstruktur, dalam *OOP* setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Objek-Oriented Programming atau Pemrograman berorientasi objek (OOP) adalah sebuah pendekatan untuk pengembangan perangkat lunak dimana struktur dari software ini didasarkan pada objek berinteraksi dengan satu sama lain untuk menyelesaikan tugas. Interaksi ini mengambil bentuk pengiriman pesan bolak-balik antara objek. Dalam menanggapi pesan, obyek dapat melakukan suatu tindakan atau metode (Clark, 2011).

Berikut dibawah ini merupakan beberapa konsep dasar dan istilah umum yang digunakan dalam program berorientasi objek:

1. Objek

   Bentuk baik yang nyata atau tidak, seperti manusia, hewan, benda, konsep, aliran, dan lain-lain. Objek merupakan inisiasi (turunan langsung) dari suatu kelas. Dalam hal OOP, objek adalah sebuah struktur yang menggabungkan sebuah data, dan prosedur untuk bekerja dengan data. Misalnya, jika Anda tertarik dalam pelacakan data yang terkait dengan produk dalam persediaan, Anda akan membuat objek produk yang bertanggung jawab untuk menjaga dan bekerja dengan data yang berkaitan dengan produk

2. Kelas

   Kumpulan objek yang memiliki kemiripan perilaku (method), cirri atau karakteristik (property). Contoh objek orang dari kelas manusia, potongan sebagai berikut: Manusia orang1=new manusia(“Parto”);

3. Method

   Perilaku dari objek atau kelas tertentu. Merupakan perwujudan aksi atau tindakan dari dunia nyata di dalam pemrograman komputer.

4. Konstruktor

   Suatu fungsi yang dideklarasikan atau didefinisikan di dalam kelas, konstruktor harus mempunyai nama yang sama dengan fungsinya.

Konstruktor dijalankan bersamaan dengan terciptanya kelas tersebut. Dalam suatu kelas bias terdapat lebih dari satu konstruktor. Konstruktor seperti method tetapi tidak mengembalikan nilai dan dapat didefinisikan tanpa parameter atau memakainya.

5. De-konstruktor

   Fungsi yang dideklarasikan dalam kelas, nama sama dengan nama fungsinya. Tetapi dijalankan bersamaan dengan dimusnahkannya kelas tersebut

6. Karakteristik / properties

   Ciri yang dimiliki oleh suatu objek, karakteristik ini juga sebagai pembeda objek satu dengan objek lainnya dalam kelas yang sama (konsep individu).

7. Variabel

   Tempat menampung data sementara, dalam pemrograman objek biasanya disebut data, sedangkan dalam pemrograman prosedural sering disebut dengan variabel

8. Data

   Istilah lain dari variabel dalam OOP. Dalam pemrograman java bisa juga disebut field, data member atau instance variable

9. Abstraksi

   Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan

10. Enkapsulasi

    Enkapsulasi adalah proses dimana tidak ada atau tidak bolehnya akses langsung yang diberikan kepada data, melainkan tersembunyi. Jika Anda ingin mendapatkan akses ke data, Anda harus berinteraksi dengan objek yang bertanggung jawab untuk data. Contoh misalkan, jika Anda ingin melihat atau memperbarui informasi tentang produk, Anda harus bekerja melalui obyek produk. Untuk membaca data, Anda akan mengirim objek produk pesan. Obyek produk kemudian akan membaca nilai dan mengirim kembali pesan yang mengatakan kepada Anda apakah nilai tersebut. Obyek Produk mendefinisikan operasi apa yang dapat dilakukan pada data produk. Jika Anda mengirim pesan untuk memodifikasi data dan objek produk menentukan itu adalah permintaan yang valid, maka

operasi tersebut akan dilakukan untuk Anda dan mengirim pesan kembali dengan hasilnya.

11. Polimorfisme

    Polimorfisme adalah kemampuan dari dua objek yang berbeda untuk merespon pesan permintaan yang sama dengan cara mereka sendiri yang unik. Sebagai contoh, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut.

12. Inheritance

    Inheritance atau warisan digunakan dalam OOP untuk mengklasifikasikan objek dalam suatu program yang sesuai dengan karakteristik dan fungsi umum. Hal ini membuat bekerja dengan objek-objek menjadi lebih mudah dan lebih intuitif. Hal tersebut juga membuat *programming* menjadi lebih mudah karena memungkinkan untuk menggabungkan karakteristik umum menjadi objek *parent* dan mewarisi karakteristik tersebut ke dalam objek *child.*

13. Agregasi

    Agregasi adalah ketika sebuah objek terdiri dari gabungan dari benda-benda lain yang bekerja sama. Misalnya, rumput mesin pemotong objek Anda adalah gabungan dari objek roda, obyek mesin, obyek pisau, dan sebagainya. Bahkan, objek mesin adalah gabungan dari berbagai obyek. Kemampuan untuk menggunakan agregasi dalam OOP merupakan salah satu fitur canggih yang memungkinkan Anda untuk secara akurat membuat model dan mengimplementasikan proses bisnis dalam program Anda.

14. Hak akses *(access attribute)*

    Hak akses digunakan untuk dapat menentukan data member mana yang dapat digunakan oleh kelas lain, dan mana yang tidak dapat digunakan. Hak akses ini sangat penting dalam membuat program turunan kelas.

    a. *Public*

       Data member atau variable dapat diakses dari kelas mana saja

    b. *Private*

       Kelas yang data membernya memakai private hanya dapat digunakan oleh kelas bersangkutan, tidak dapat digunakan kelas lain

    c. *Protected*

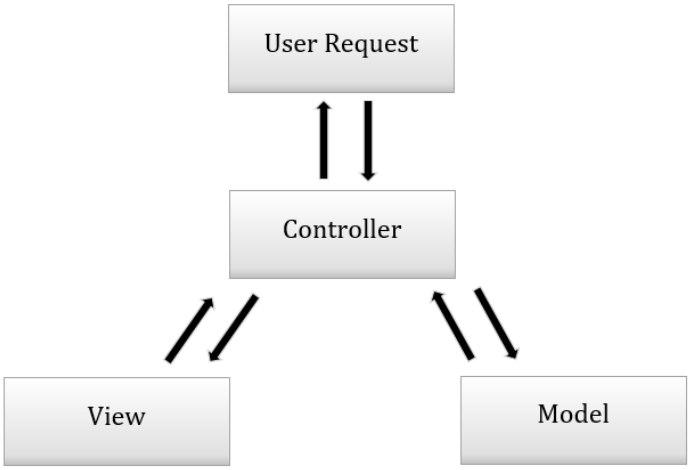
       Dapat mengakses data member dari kelas dalam package yang sama dan subkelasnya

Dengan mengunakan *OOP* maka dalam melakukan pemecahan suatu masalah kita tidak dapat melihat bagaimana cara menyelesaikan suatu masalah seperti pada pemrograman terstruktur tetapi objek-objek aoa yang dapat melakukan pemecahan masalah tersebut.

2.2.5.1 Metode MVC

Dalam dunia *Object-Oriented Programming (OOP)*. Tipe pemrograman ini adalah berbasis object. Dalam pembuatan aplikasi pada tugas akhir ini penulis menggunakan metode MVC. Prinsip utama Metode MVC adalah membagi kerja sistem antara Model, View, dan Controller dengan menggunakan framework Codeigniter. Dengan MVC, maka memungkinkan pemisahan antara layer application-logic dan presentation. Sehingga, dalam sebuah pengembangan web, seorang programmer bisa berkonsentrasi pada core-sistem, sedangkan web designer bisa berkonsentrasi pada tampilan web. Menariknya, skrip PHP, query MySQL, Javascript dan CSS bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan resource besar pula untuk mengesekusinya (Hakim, 2010).

Adapun alur program aplikasi berbasis framework Codeigniter dapat dilihat pada Gambar 2.2 (Hakim, 2010).



Gambar 2.2 Alur MVC

Sumber: Hakim (2010)

Gambar diatas menerangkan bahwa ketika datang sebuah user request, maka akan ditangani oleh controller, kemudian controller akan memanggil model jika memang diperlukan operasi database. Hasil dari query oleh model kemudian akan dikembalikan ke controller. Selanjutnya controller akan memanggil view yang tepat dan mengkombinasikannya dengan hasil query model. Hasil akhir dari operasi ini akan ditampilkan dibrowser.

Dalam konteks CodeIgniter dan aplikasi berbasis web, maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu :

1. Model

   Kode program (berupa OOP class) yang digunakan untuk memanipulasi database.

2. View

   Berupa template html/xml atau php untuk menampilkan data pada browser

3. Controller

   Kode program (berupa OOP class) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontol model dan View)

2.2.5.2 CodeIgniter

CodeIgniter adalah sebuah framework PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi web berbasis PHP dibanding jika menulis semua kode program dari awal (Hakim, 2010).

CodeIgniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. (http://ellislab.com), sebuah perusahaan yang memproduksi CMS *(Content Management Sistem)* yang cukup handal, yaitu Expression Engine (http://www.expressionengine.com). Saat ini, CodeIgniter dikembangkan dan dimaintain oleh *Expression Engine Development Team*.

Adapun beberapa keuntungan menggunakan CodeIgniter, diantaranya:

1. Gratis

   CodeIgniter berlisensi dibawah Apache/BSD opensorce.

2. Ditulis Menggunakan PHP 4

   Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.

3. Berukuran Kecil

   Ukuran CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan framework lain yang berukuran besar.

4. Menggunakan Konsep MVC

   CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan layer application-logic dan presentation.

5. URL yang Sederhana

   Secara default, URL yang dihasilkan CodeIgniter sangat bersih dan *Serach Engine Friendly (SEF)*.

6. Memiliki Paket Library yang Lengkap

   CodeIgniter mempunyai library yang lengkap untuk mengerjakan operasioperasi yang umum dibutuhkan oleh sebuah aplikasi berbasis

web, misalnya mengakses database, mengirim email, memvalidasi form, menangani session dan sebagainya.

7. *Extensible*

   Sistem dapat dikembangkan dengan mudah menggunakan plugin dan helper, atau dengan menggunakan hooks.

8. Tidak Memerlukan Template Engine

   Meskipun CodeIgniter dilengkapi dengan template parser sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

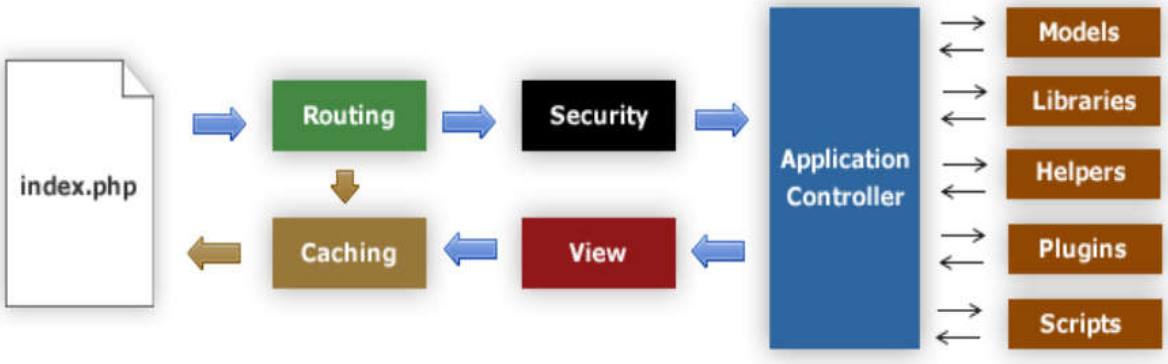
9. Dokumentasi Lengkap dan Jelas

   Dari sekian banyak framework, CodeIgniter adalah satu-satunya framework dengan dokumentasi yang lengkap dan jelas.

10. Komunitas

    Komunitas CodeIgniter saat ini berkembang pesat. Salah satu komunitasnya bisa dilihat di (http://codeigniter.com/forum/).

Proses aliran data aplikasi pada sistem dapat diilustrasikan seperti terlihat pada Gambar 2.3



Gambar 2.3 *Codeigniter Application Flowchart*

Sumber: Userguide 3 Codeigniter (2017)

Keterangan :

1. Index.php berfungsi sebagai front controller, menginisialisasi base resorce untuk menjalankan CodeIgniter.
2. Router memerikasa HTTP request untuk menentukan apa yang harus dilakukan dengannya.
3. Jika Cache aktif, maka hasilnya akan langsung dikirimkan ke browser dengan mengabaikan aliran data normal.
4. Security. Sebelum Controller dimuat, HTTP request dan data yang dikirimkan user akan difilter untuk keamanan.

5. Controller memuat model, core libraries, plugins, helpers dan semua resource yang diperlukan untuk memproses request.
6. Akhirnya View yang dihasilkan akan dikirimkan ke browser. Jika Cache aktif, maka View akan disimpan sebagai Cache dahulu, sehingga pada request berikutnya langsung dapat ditampilkan

2.2.6 Pengujian Perangkat Lunak

Untuk mengurangi atau menghilangkan kesalahan pada perangkat lunak diperlukan suatu tahap pengujian untuk menemukan kesalahan-kesalahan yang ada pada perangkat lunak. Pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap *item* perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada setiap komponen serta mengevaluasi fasilitas-fasilitas dari perangkat lunak yang akan dikembangkan. Tujuan dari pengujian adalah untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program sebelum menyerahkan program kepada *customer*. Salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas tinggi dalam menemukan kesalahan (Pressman, 2010)

Beberapa aturan yang dapat digunakan sebagai penjelasan tentang pengujian perangkat lunak: (Myers, G,2004)

1. Pengujian merupakan sebuah proses proses eksekusi program dengan tujuan utama yaitu untuk mencari kesalahan
2. Sebuah kasus pengujian dikatakan baik jika memiliki kemungkinan penemuan kesalahan yang tinggi
3. Pengujian yang berhasil adalah pengujian yang menemukan kesalahan

Berdasarkan ketiga pernyataan diatas dapat disimpulkan bahwa pengujian yang baik tidak hanya ditujukan untuk menemukan kesalahan pada perangkat lunak tetapi juga untuk dapat ditemukannya data uji yang dapat menemukan kesalahan secara lebih teliti dan cepat.

2.2.7 Metode Pengujian Perangkat Lunak

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak. Metode pengujian berhubungan dengan perancangan data uji yang akan dieksekusi pada perangkat lunak yang dikembangkan. Metode pengujian diharapkan mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap *(completeness of test)* dan mempunyai kemungkinan tinggi untuk menemukan kesalahan *(high likelihood for uncovering error).* Adapun metode pengujian perangkat lunak dapat dilakukan dengan dua cara yaitu:

2.2.7.1 Pengujian Kotak Hitam *(Black Box Testing)*

Pengujian ini dilakukan dengan mengeksekusi data uji dan mengecek apakah setiap fungsional perangkat lunak dapat bekerja dengan benar. Data uji

diperoleh dari spesifikasi kebutuhan perangkat lunak, yang dalam hal ini menjelaskan kebutuhan fungsional perangkat lunak.

Black-Box testing berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program. *Black-Box testing* berusaha untuk menemukan kesalahan dalam kategori berikut: (Pressman, 2010)

1. Fungsi yang tidak benar atau fungsi yang hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan perilaku (behavior) atau kesalahan kinerja
5. Inisialisasi dan pemutusan kesalahan

2.2.7.2 Pengujian Kotak Putih *(White Box Testing)*

Pengujian ini dilakukan dengan menggunakan data uji untuk menguji semua elemen program perangkat lunak (data internal, proses logika keputusan, perulangan/*loop,* jalur). Data uji diperoleh dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.

2.2.8 Sistem Database

Menurut Elmasri dan Navathe (2011), *database* adalah kumpulan-kumpulan data-data yang saling terkait. Definisi *database* yang cukup umum, misalnya *database* adalah kumpulan data yang logis dan koheren dengan beberapa makna yang melekat. Beraneka ragam data yang *random* tidak bisa disebut sebagai *database.* Sebuah *database* dirancang, dibangun, dan diisi dengan data untuk tujuan tertentu. *Database* ini memiliki kelompok yang dimaksudkan pada pengguna dan beberapa aplikasi yang terbentuk sebelumnya di mana para pengguna tertarik.

Menurut Connolly dan Begg (2010:54), sistem Database adalah kumpulan program aplikasi yang berinteraksi dengan Database, bersamaan dengan Database Management Sistem (DBMS) dan Database itu sendiri.

2.2.8.1 Database Management Sistem (DBMS)

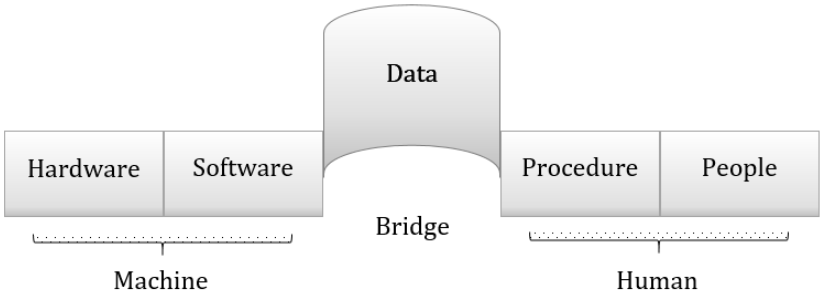
Pemahaman mengenai Database Management Sistem (DBMS) akan dijabarkan menjadi definisi DBMS, komponen DBMS, dan fungsi DBMS.

A. Definisi DBMS

   Menurut Connolly dan Begg (2010:66), DBMS adalah sistem perangkat lunak yang mengizinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke dalam Database.

B. Komponen DBMS

DBMS terdiri dari lima komponen utama, yaitu hardware, software, data, procedures, dan people. Gambar 2.4 menggambarkan lima komponen utama DBMS. (Connolly & Begg, 2010)



Gambar 2.4 Komponen Utama DBMS

Sumber: Connolly & Begg (2010)

Keterangan:

- *Hardware* (Perangkat Keras)

  Suatu DBMS menggunakan perangkat keras untuk menjalankan aplikasinya. Perangkat keras yang digunakan dapat berupa Personal Computer (PC), Mainframe, jaringan komputer.

- *Software* (Perangkat Lunak)

  Komponen perangkat lunak terdiri dari perangkat lunak DBMS itu sendiri dan program aplikasi beserta dengan sistem operasinya, serta perangkat lunak jaringan apabila DBMS digunakan dalam sebuah jaringan.

- Data

  Dari sudut pandang pengguna, data merupakan komponen terpenting dalam suatu DBMS. Data bertindak sebagai jembatan penghubung antara komponen mesin dengan komponen manusia.

- *Procedure*

  Instruksi-instruksi dan aturan-aturan yang digunakan dalam merancang dan menggunakan suatu Database.

- *People* (Manusia)

  Komponen terakhir dalam lingkungan DBMS adalah manusia, dimana merupakan komponen yang terlibat langsung dengan sistem

C. Fungsi DBMS

Menurut Connolly dan Begg (2010:66), DBMS memiliki beberapa fungsi sebagai berikut:

1. Mengizinkan pengguna untuk mendefinisikan Database melalui Data Definition Language (DDL). DDL mengizinkan pengguna

menspesifikasikan tipe data, struktur, dan aturan data yang disimpan ke dalam Database.

2. Mengizinkan pengguna untuk melakukan operasi insert, update, delete, dan retrieve data dari Database melalui Data Manipulation Language (DML)
3. Menyediakan akses kontrol ke dalam Database, yang meliputi:
    - Sistem keamanan, dimana mencegah pengguna yang tidak memiliki izin untuk mengakses Database.
    - Sistem integritas, dimana memelihara konsistensi data yang tersimpan.
    - Konkurensi sistem kontrol, dimana mengizinkan akses Database yang dapat digunakan bersama.
    - Sistem kontrol pemulihan, dimana Database dapat mengembalikan keadaan data seperti sebelumnya apabila terjadi kegagalan perangkat lunak atau perangkat keras.
    - Katalog yang dapat diakses pengguna, dimana mengandung penjelasan tentang data yang ada di dalam Database.

2.2.8.2 Structured Query Language (SQL)

SQL adalah bahasa *database* yang komprehensif yaitu memiliki pernyataan untuk mendefinisikan data, *query*, dan *update*. Awalnya, SQL atau *Structured Query Language* disebut SEQUEL *(Structured Query Language English)* dan dirancang dan diterapkan di IBM Research sebagai antarmuka untuk eksperimental sistem *database* relasional yang disebut SISTEM R.SQL sekarang merupakan bahasa standar untuk DBMS relasional komersial (Elmasri & Navathe, 2011)

Menurut Connolly dan Begg (2010), SQL adalah transformoriented languange atau bahasa yang didesain untuk menggunakan relasi yang mentrasformasikan input menjadi output yang dibutuhkan. Sebagai bahasa standar ISO SQL memiliki dua komponen utama, yaitu:

a. Data Definition Language (DDL)

    DDL adalah bahasa yang mengizinkan *Database Administrator* (DBA) atau pengguna untuk menjelaskan dan menamakan entitas, atribut, dan relasi yang dibutuhkan aplikasi, beserta integritas dan aturan kemanan.

b. Data Manipulation Language (DML)

    DML adalah bahasa yang menyediakan kumpulan operasi untuk mendukung operasi manipulasi data dalam Database.

2.2.9 Entity Relational Modelling (ER Modelling)

Menurut Connolly dan Begg (2010), *ER Modelling* merupakan model yang dapat digunakan untuk mendapatkan pengertian tentang data yang akan digunakan perusahaan dan bagaimana penggunaannya. Aspek utama dari *ER Modelling* yaitu:

1. *Entity Type*

   Entity Type adalah kumpulan objek-objek dengan property yang sama dimana diidentifikasi dengan perusahaan yang keberadaanya independent.

2. *Relationship Type*

   Relationship Type adalah sekumpulan hubungan antar tipe entitas yang memiliki arti. *Relationship Occurrence* adalah sebuah hubungan yang dapat diidentifikasikan secara unik yang meliputi satu kejadian dari setiap entitas yang ada.

3. *Attribute*

   Attribute adalah sebuah kolom yang memiliki nama pada sebuah relasi. Dalam *relational model*, relasi digunakan untuk menyimpan informasi dari objek yang akan direpresenstasikan dalam database. Relasi merepresentasikan tabel dua dimensi, dimana baris dari table sesuai dengan individual records dan kolom pada tabel sesuai dengan attribute (Connolly dan Begg, 2010).

4. *Keys*

   Ada lima jenis *keys* yang biasa digunakan dalam *ER Modeling*, yaitu:

   a. *Candidate Key*

      Candidate key merupakan sejumlah kecil atribut dari entitas yang dapat mengidentifikasikan setiap kejadian dari entitas tersebut secara unik.

   b. *Primary Key*

      Primary key merupakan *candidate key* yang dipilih untuk mengidentifikasikan setiap kejadian dari entitas secara unik. Pemilihan primary key pada suatu entitas didasarkan pada pertimbangan panjang atribut, jumlah minimal atribut yang dibutuhkan, serta tingkat keunikannya.

   c. *Alternate Key*

      Alternate key merupakan kumpulan *candidate key* yang tidak terpilih menjadi *primary key*.

   d. *Composite Key*

Composite key merupakan *candidate key* yang terdiri dari dua atau lebih atribut.

e. *Foreign Key*

Foreign key merupakan sebuah atribut atau sekumpulan atribut pada suatu relasi yang sama atau sesuai dengan *candidate key* dari relasi lainnya

2.2.10 Metode *Analytical Hierarchy Process (AHP)*

AHP merupakan sistem pembuat keputusan dengan menggunakan model matematis. AHP membantu dalam menentukan prioritas dari beberapa kriteria dengan menggunakan model matematis. AHP membantu dalam menentukan prioritas dari beberapa kriteria dengan melakukan Analisis perbandingan berpasangan dari masing-masing kriteria (Hanafi, 2011).

2.2.10.1 Prinsip-prinsip Dasar *AHP*

Dalam menyelesaikan permasalahan dengan metode *AHP* tentunya ada beberapa prinsip yang harus dipahami, diantaranya: (Kusrini, 2007)

1. Membuat Hirarki

Sistem yang kompleks bisa dipahami dengan memecah sistem tersebut menjadi elemen-elemen pendukung, menyusun elemen secara hirarki, dan menggabungkannya atau mensistensikannya.

2. Menentukan Kriteria dan Alternatif

AHP melakukan perbandingan berpasangan antar dua elemen pada level yang sama. Kedua elemen tersebut dibandingkan dengan menimbang tingkat preferensi elemen yang satu terhadap elemen yang lain berdasarkan kriteria tertentu.

Tabel 2.6 Skala Penilaian Perbandingan Berpasangan

Intensitas Kepentingan	Keterangan
1	Kedua elemen sama pentingnya
3	Elemen yang satu sedikit lebih penting daripada elemen yang lainnya
5	Elemen yang satu lebih penting daripada elemen yang lainnya
7	Satu elemen jelas lebih mutlak penting daripada elemen yang lainnya
9	Satu elemen mutlak penting daripada elemen yang lainnya

2,4,6,8	Nilai-nilai antara dua nilai pertimbangan yang berdekatan
Kebalikan	Jika aktivitas ke-i mendapat satu angka bila dibandingkan dengan aktivitas ke-j, maka j memiliki nilai kebalikannya dibandingkan dengan i.

3. Menentukan Prioritas

Untuk setiap kriteria dan alternative, harus dilakukan perbandingan berpasangan *(pairwise comparisons)*. Nilai-nilai perbandingan relative dari seluruh alternative kriteria bisa disesuaikan dengan judgement yang sudah ditentukan untuk menghasilkan bobot dan prioritas. Bobot dan prioritas tersebut kemudian dihitung dengan memanipulasi matriks atau melalui penyelesaian persamaan matematika.

4. Konsistensi Logis

Konsistensi memiliki dua arti. Pertama, objek-objek yang serupa bisa dikelompokkan sesuai dengan keseragaman dan relevansi. Kedua, menyangkut tingkat hubungan antara objek yang didasarkan pada kriteria tertentu.

2.2.10.2 Langkah-langkah Penggunaan Metode *AHP*

Langkah-langkah dalam penggunaan metode *AHP* (menurut Kusrini, 2007) meliputi:

1. Mendefinisikan masalah dan menentukan solusi yang diharapkan, kemudian menyusun hirarki dari permasalahan yang dihadapi. Penyusunan hirarki dilakukan dengan menetapkan tujuan yang merupakan target sistem secara keseluruhan.
2. Menentukan prioritas elemen
    a. Langkah pertama dalam menentukan prioritas elemen adalah membuat perbandingan pasangan, yaitu membandingkan elemen secara berpasangan sesuai kriteria yang diberikan
    b. Matriks perbandingan berpasangan di isi menggunakan bilangan untuk mempresentasikan kepentingan relatif dari suatu elemen terhadap elemen yang lainnya
3. Sintesis

    Pertimbangan-pertimbangan terhadap perbandingan berpasangan disintesis untuk memperoleh keseluruhan prioritas. Hal-hal yang dilakukan dalam langkah ini adalah:

    a. Menjumlahkan nilai-nilai dari setiap kolom pada matriks
    b. Membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks.

c. Menjumlahkan nilai-nilai dari setiap baris dan membaginya dengan jumlah elemen untuk mendapatkan nilai rata-rata

4. Mengukur konsistensi

   Dalam pembuatan keputusan, penting untuk mengetahui seberapa baik konsistensi yang ada karena kita tidak menginginkan keputusan berdasarkan pertimbangan dengan konsistensi yang rendah. Hal-hal yang dilakukan dalam langkah ini:

   a. Kalikan setiap nilai pada kolom pertama dengan prioritas relative elemen pertama, nilai pada kolom kedua dengan prioritas relatif elemen kedua, dan seterusnya.
   b. Jumlahkan setiap baris
   c. Hasil dari penjumlahan baris dijumlahkan dengan elemen prioritas relatif yang bersangkutan.
   d. Jumlahkan hasil penjumlahan diatas dengan banyaknya elemen yang ada, hasilnya disebut maks.

5. Hitung *consistency index (CI)* dengan rumus:

   $$CI = (\lambda maks\text{-}n) / n\text{-}1 \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

   Dimana n = banyaknya elemen

6. Hitung rasio konsistensi/consistency ratio (CR) denga rumus:

   $$CR = CI / IR \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

   Dimana CR = *consistency ratio*

   CI = *consistency index*

   IR = *indeks random consistency*

7. Memeriksa sama dengan konsistensi hierarki. Jika nilainya lebih dari 10%, maka penilaian data judgement harus diperbaiki. Namun jika rasio konsistensi (CI / IR) kurang atau sama dengan 0,1, maka hasil perhitungan bisa dinyatakan benar. Daftar indeks random konsistensi (IR) bisa dilihat pada tabel 2.2 berikut:

Tabel 2.7 Daftar indeks Random Konsistensi (IR)

Ukuran Matriks	Nilai IR
1.2	0.00
3	0.58
4	0.90
5	1.12
6	1.24
7	1.32

8	1.41
9	1.45
10	1.49
11	1.51
12	1.48
13	1.56
14	1.57
15	1.59

2.2.10.3 Pembobotan Nilai Kriteria dan Alternatif

Setelah menentukan kriteria-kriteria untuk penentuan lokasi usaha berdasarkan sektor, selanjutnya adalah pembobotan nilai antar kriteria dari masing-masing sektor usaha. Berikut adalah skenario rancangan pembobotan nilai antar kriteria:

1. Sektor Pertanian

Tabel 2.8 Skenario Rancangan Pembobotan Kriteria Sektor Pertanian

Kriteria	Luas Lahan	Komoditi Kluster	Bantuan Alat
Luas Lahan	1	3	7
Komoditi Kluster	0,33	1	5
Bantuan Alat	0,14	0,20	1

Keterangan:

- Kriteria Luas Lahan sama pentingnya dengan kriteria Luas Lahan
- Kriteria Luas Lahan sedikit lebih penting daripada kriteria Komoditi Luas Lahan
- Kriteria Luas Lahan sangat penting daripada kriteria Bantuan Alat
- Kriteria Komditi Kluster lebih penting daripada kriteria Bantuan Alat

2. Sektor Kesehatan

Tabel 2.9 Skenario Rancangan Pembobotan Kriteria Sektor Kesehatan

Kriteria	Jumlah Penduduk	Tingkat Ekonomi	Jenis Pelayanan Sekitar
Jumlah Penduduk	1	5	7
Tingkat Ekonomi	0,20	1	3
Jenis Pelayanan Sekitar	0,14	0,33	1

Keterangan:

- Kriteria Jumlah Penduduk sama pentingnya dengan kriteria Jumlah Penduduk
- Kriteria Jumlah Penduduk sedikit lebih penting daripada kriteria Tingkat Ekonomi
- Kriteria Jumlah Penduduk sangat penting daripada kriteria Jenis Pelayanan Sekitar
- Kriteria Tingkat Ekonomi sedikit lebih penting daripada kriteria Jenis Pelayanan Sekitar

Kemudian untuk daftar alternatif yang sudah didapatkan juga dilakukan pembobotan nilai antar alternatif

1. Sektor Pertanian

   Berdasarkan data statistik mengenai luas lahan pertanian yang dimiliki tiap kecamatan di kabupaten malang, maka skenario pembobotan nilai dilakukan sebagai berikut:

Tabel 2.10 Skenario Rancangan Pembobotan Kriteria Sektor Pertanian

Kriteria : Luas Lahan	Donomulyo	Kepanjen	Kalipare
Donomulyo	1	3	5
Kepanjen	0,33	1	7
Kalipare	0,20	0,14	1

Keterangan:

- Donomulyo sama pentingnya dengan Donomulyo berdasarkan kriteria Luas Lahan
- Donomulyo sedikit lebih penting daripada Kepanjen berdasarkan kriteria Luas Lahan
- Donomulyo lebih penting daripada Kalipare berdasarkan kriteria Luas Lahan
- Kepanjen sangat penting daripada Kalipare berdasarkan kriteria Luas Lahan

2. Sektor Kesehatan

   Berdasarkan data statistik mengenai jumlah penduduk yang dimiliki masing-masing kecamatan di kabupaten malang, maka skenario pembobotan nilai dilakukan sebagai berikut:

Tabel 2.11 Skenario Rancangan Pembobotan Kriteria Sektor Kesehatan

Kriteria : Jumlah Penduduk	Donomulyo	Kepanjen	Kalipare

Donomulyo	1	3	7
Kepanjen	0,33	1	9
Kalipare	0,14	0,11	1

Keterangan:

- Donomulyo sama pentingnya dengan Donomulyo berdasarkan kriteria Jumlah Penduduk
- Donomulyo sedikit lebih penting daripada Kepanjen berdasarkan kriteria Jumlah Penduduk
- Donomulyo sangat penting daripada Kalipare berdasarkan kriteria Jumlah Penduduk
- Kepanjen mutlak sangat penting daripada Kalipare berdasarkan kriteria Jumlah Penduduk

Pada tahap ini akan menjelaskan mengenai salah satu tahap dalam melakukan proses analisis potensi usaha menggunakan *AHP* yaitu bagaimana pengguna memberikan penilaian atau pembobotan nilai antar kriteria dan alternatif. Sesuai dengan teori dasar *AHP* yang telah dijelaskan pada bab sebelumnya pembobotan nilai memiliki skala penilaian mulai dari angka 1-9 dan dilakukan oleh penganalisa berdasarkan pengetahuan yang cukup dimiliki oleh pengguna. Skala penilaian ini dalam teori *AHP* disebut sebagai nilai preferensi (tingkat kepentingan) antar kriteria dan alternatif.

2.2.11 Peta dan Pemetaan

Permanasari (Said, 2013) menyatakan bahwa peta adalah suatu representasi atau gambaran unsur-unsur atau kenampakan abstrak yang dipilih dari permukaan bumi atau benda-benda angkasa dan umumnya digambarkan pada satu bidang datar dan diperkecil atau diskalakan.

Pemetaan adalah pengelompokkan suatu kumpulan wilayah yang berkaitan dengan beberapa letak geografis wilayah yang meliputi dataran tinggi, pegunungan, sumber daya dan potensi penduduk yang berpengaruh terhadap sosial kultural yang memilki ciri khas khusus dalam penggunaan skala yang tepat (Dalimunthe, 2008).

Dengan adanya peta dan pemetaan berdasarkan lokasi serta informasi sumber daya lahan di Kabupaten Malang, maka masyarakat atau investor mendapatkan informasi secara efektif dan akurat mengenai potensi-potensi usaha yang ada di daerah tersebut.

2.2.12 *Geographic Information Sistem (GIS)*

Menurut Raper J (dalam Prahasta, 2009) menyatakan bahwa *GIS* adalah sistem yang dapat mendukung (proses) pengambilan keputusan (terkait aspek) spasial dan mampu mengintegrasikan deskripsi-deskripsi lokasi dengan

karakteristik-karakteristik fenomena yang ditemukan di lokasi tersebut. *GIS* yang lengkap akan mencakup metodologi dan teknologi yang diperlukan; yaitu, data spasial, perangkat keras, perangkat lunak dan struktur organisasi.

GIS mampu mengakomodasi penyimpanan, pemrosesan, dan penayangan data spasial digital bahkan integrasi data yang beragam, mulai dari citra satelit, foto udara, peta bahkan data statistik.

2.2.12.1 Subsistem *GIS*

GIS dapat diuraikan menjadi beberapa sub-sub sistem sebagai berikut: (Prahasta, 2009)

1. Data Input : Subsistem ini bertugas untuk mengumpulkan, mempersiapkan, dan menyimpan data spasial dan atributnya dari berbagai sumber. Sub-sistem ini pula yang bertanggung jawab dalam mengonversikan atau mentransformasikan format -format data aslinya ke dalam format yang dapat digunakan oleh perangkat SIG yang bersangkutan
2. Data Output : Sub-sistem ini bertugas untuk menampilkan atau menghasilkan keluaran (termasuk mengekspornya ke format yang dikehendaki) seluruh atau sebagian Database (spasial) baik dalam bentuk softcopy maupun hardcopy seperti halnya tabel, grafik, report, peta, dan lain sebagainya
3. Data Management :Sub-sistem ini mengorganisasikan baik data spasial maupun tabel-tabel atribut terkait ke dalam sebuah sistem Database sedemikian rupa hingga mudah dipanggil kembali atau di-retrieve, diupdate, dan diedit
4. Data Manipulation & Analysis Sub-sistem ini menentukan informasi informasi yang dapat dihasilkan oleh SIG. Selain itu sub-sistem ini juga melakukan manipulasi (evaluasi dan penggunaan fungsi- fungsi dan operator matematis & logika) dan pemodelan data untuk menghasilkan informasi yang diharapkan

2.2.12.2 Komponen *GIS*

Geographic Information Sistem (GIS) terdiri dari beberapa komponen dengan berbagai karakteristiknya antara lain: (Prahasta, 2009)

1. Perangkat Keras.

   Adapun perangkat keras yang sering digunakan untuk aplikasi SIG adalah komputer (PC), mouse, monitor (plus VGA-card grafic) yang beresolusi tinggi, digitizer, printer, plotter, receiver GPs dan scanner
2. Perangkat Lunak.

   Dari sudut pandang yang lain, SIG bisa juga merupakan sistem perangkat lunak yang tersusun secara modular di mana sistem Databasenya memegang peranan kunci.
3. Data dan Informasi Goegrafi.

   SIG dapat mengumpulkan dan menyimpan data atau informasi yang diperlukan baik secara tidak langsung (dengan cara meng-import-nya dari format-format perangkat lunak SIG yang lain) maupun secara langsung dengan cara melakukan dijitasi data spasialnya di atas tampilan layar monitor, atau manual dengan menggunakan peta analog dan kemudian memasukkan data atributnya dari tabel-tabel atau laporan dengan menggunakan keyboard
4. Manajemen.

Suatu proyek SIG akan berhasil jika di kelola dengan baik dan di kerjakan oleh orang-orang yang memiliki keakhlian (kesesuaian dengan job-description yang bersangkutan) yang tepat pada semua tingkatan.

2.2.12.3 Kemampuan *GIS*

Sistem informasi geografis mempunyai kemampuan menghubungkan berbagai data pada suatu titik tertentu di bumi, menggabungkannya, menganalisis dan akhirnya memetakan hasilnya. Adapun rincian kemampuan *GIS* dapat diuraikan sebagai berikut: (Prahasta, 2009)

1. Memasukkan dan mengumpulkan data geografis (spasial dan atribut)
2. Mengintegrasikan data geografis.
3. Memeriksa, meng-update (meng-edit) data geografis.
4. Menyimpan atau memanggil kembali data geografis.
5. Mempresentasikan atau menampilkan data geografis.
6. Mengelola, memanipulasi dan menganalisis data geografis.
7. Menghasilkan output data geografis dalam bentuk peta tematik (view dan layout), tabel, grafik (chart) laporan, dan lainnya baik dalam bentuk hardcopy maupun softcopy

2.2.12.4 Aplikasi WEB berbasis *GIS*

Sistem ini merupakan aplikasi yang berjalan pada media jaringan LAN dan atau internet; khususnya dengan menggunakan layanan web-nya. Aplikasi web-based GIS hanya membantu para penggunanya dalam proses menginternetkan peta-peta dijitalnya (baik format raster maupun vektor) sedemikian rupa sehingga dapat diakses oleh berbagai komunitas yang memakai program aplikasi brows internet (Prahasta, 2009)

Kenna (dalam Prahasta, 2009) menyatakan bahwa saat ini, web-base GIS semakin menarik untuk diintegrasikan pada aplikasi web yang lebih luas lagi (content management sistem/CMS) agar website yang terbentuk juga memiliki layanan-layanan terkait dengan tampilan dan analisis spasial yang sangat menarik.

2.2.13 Google Maps API

Google Maps adalah layanan Google yang diperkenalkan pada Februari 2005. Google Maps menawarkan teknologi pemetaan yang bersifat user-friendly dan berisi informasi bisnis local yang mencakup lokasi bisnis, informasi kontak dan arah perjalanan. Cara menggunakan Google Maps pun mudah, pengguna hanya perlu menggeser atau menarik peta tersebut untuk dapat menavigasinya. Google Maps berjalan dengan menggunakan HTML, CSS, dan Javascript yang saling bekerja sama.

Sebelum adanya public *Application Programming Interface (API)*, seorang hacker membobol atau meng-hack Google Maps untuk dimasukkan atau di integrasikan ke suatu website, karena itulah dibuat API yang bertujuan untuk membuat Google Maps API tersebut menjadi free akan tetapi google bebas

untuk memasukkan advertisement ke Google Maps API tersebut (Svennerberg, 2010).

Selama beberapa tahun terakhir, *Google Earth* dan *Google Maps* telah digunakan oleh banyak lembaga akademis sebagai penelitian akademik dan alat pemetaan. Sebuah survei dilakukan untuk menentukan popularitas produk pemetaan, dan jenis penggunaan dalam suasana perpustakaan akademik. Hasil penelitian menunjukkan bahwa lebih dari 90 persen responden menggunakan *Google Earth* dan *Google Maps* baik untuk membantu menjawab pertanyaan penelitian, ataupun untuk membuat dan mengakses dalam menemukan alat bantu, dengan tujuan instruksional atau untuk promosi dan pemasaran (Dodsworth & Nicholson, 2012)